

On A Semi-Automatic Method for Generating Composition Tables

Weiming Liu and Sanjiang Li

Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia

Abstract. Originating from Allen’s Interval Algebra, composition-based reasoning has been widely acknowledged as the most popular reasoning technique in qualitative spatial and temporal reasoning. Given a qualitative calculus (i.e. a relation model), the first thing we should do is to establish its composition table (CT). In the past three decades, such work is usually done manually. This is undesirable and error-prone, given that the calculus may contain tens or hundreds of basic relations. Computing the correct CT has been identified by Tony Cohn as a challenge for computer scientists in 1995. This paper addresses this problem and introduces a semi-automatic method to compute the CT by randomly generating triples of elements. For several important qualitative calculi, our method can establish the correct CT in a reasonable short time. This is illustrated by applications to the Interval Algebra, the Region Connection Calculus RCC-8, the INDU calculus, and the Oriented Point Relation Algebras. Our method can also be used to generate CTs for customised qualitative calculi defined on restricted domains.

1 Introduction

Since Allen’s seminal work of Interval Algebra (IA) [1,2], qualitative calculi have been widely used to represent and reason about temporal and spatial knowledge. In the past decades, dozens of qualitative calculi have been proposed in the artificial intelligence area “Qualitative Spatial & Temporal Reasoning” and Geographic Information Science. Except IA, other well known binary qualitative calculi include the Point Algebra [3], the Region Connection Calculi RCC-5 and RCC-8 [4], the INDU calculus [5], the Oriented Point Relation Algebras \mathcal{OPRA} [6], and the Cardinal Direction Calculus (CDC) [7], etc.

Relations in each particular qualitative calculus are used to represent temporal or spatial information at a certain granularity. For example, The Netherlands is *west* of Germany, The Alps *partially overlaps* Italy, I have today an appointment with my doctor *followed by* a check-up.

Given a set of qualitative knowledge, new knowledge can be derived by using constraint propagation. Consider an example in RCC-5. Given that The Alps *partially overlaps* Italy and Switzerland, and Italy is a *proper part of* the European Union (EU), and Switzerland is *discrete from* the EU, we may infer that The Alps *partially overlaps* the EU. The above inference can be obtained by using composition-based reasoning. The composition-based reasoning technique has been extensively used in qualitative

spatial and temporal reasoning, and, when combined with backtracking methods, has been shown to be complete in determining the consistency problem for several important qualitative calculi, including IA, Point Algebra, Rectangle Algebra, RCC-5, and RCC-8. Moreover, qualitative constraint solvers have been developed to facilitate composition-based reasoning [8,9].

We here give a short introduction of the composition-based reasoning technique. Suppose \mathcal{M} is a qualitative calculus, and $\Gamma = \{v_i \gamma_{ij} v_j\}_{i,j=1}^n$ is a constraint network over \mathcal{M} . The composition-based reasoning technique uses a variant of the well-known *Path Consistency Algorithm*,¹ which applies the following updating rule until the constraint network becomes stable or an empty relation appears:

$$\gamma_{ij} \leftarrow \gamma_{ij} \cap \gamma_{ik} \circ_w \gamma_{kj}, \quad (1)$$

where $\alpha \circ_w \beta$ is the *weak composition* (cf. [10,11]) of two relations α, β in \mathcal{M} , namely the smallest relation in \mathcal{M} which contains the usual composition of α and β . Although for OPRA and some other calculi the composition-based reasoning is incomplete to decide the consistency problem, it remains a very efficient method to approximately solve the consistency problem.

The weak composition in a qualitative calculus \mathcal{M} is determined by its *weak composition table* (CT for short). Usually, the CT of \mathcal{M} is obtained by manually checking the consistency of $\{x\alpha y, y\beta z, x\gamma z\}$ for each triple of basic relations $\langle \alpha, \gamma, \beta \rangle$. When \mathcal{M} contains dozens or even hundreds of basic relations, this consistency-based method is undesirable and error-prone. [12] first noticed this problem and identified it as a challenge for computer scientists.

This problem remains a challenge today. We here consider several examples. The Interval Algebra and the RCC-8 algebra contain, respectively, 13 and 8 basic relations. Their CTs were established manually. But if a calculus contains a hundred basic relations, we need to determine the consistency of one million such basic networks. This is manually impossible. The *OPRA* calculi and the CDC are large qualitative spatial calculi that have drawn increasing interests. *OPRA_m* contains $4m \times (4m + 1)$ (i.e. 72, 156, 272 for $m = 2, 3, 4$, respectively) basic relations [6], while the CDC contains 218 basic relations [7]. Sometimes we need ingenious and special methods to establish CT for such a calculus. For the *OPRA* calculi, the algorithm presented in the original paper [6] contains gaps and errors. Later, [13] presented the second algorithm, which is quite lengthy and cumbersome. Another simple algorithm has also been proposed recently [14]. Given the huge number of basic relations of *OPRA_m*, the validity of these algorithms need further verification. As for the CDC, [7] first studied the weak composition. Later, [15] noticed errors in Goyal's method and gave a new algorithm to compute the weak composition. Unfortunately, in several cases, their algorithm does not generate the correct weak composition (see [16]).

In this paper, we respond to this challenge and propose a semi-automatic approach to generate CT for general qualitative calculi. In the remainder of this paper, we first recall basic notions and results about qualitative calculi and weak composition tables in Section 2, and then apply our method to IA, INDU, RCC-8, and *OPRA₁* and *OPRA₂*

¹ The notion of Path Consistency is usually defined for constraints on finite domains, and not always appropriate for general qualitative constraints, which are defined on infinite domains.

in Section 3. An analysis of the strength and weakness of our approach is given in Section 4. Section 5 then concludes the paper.

2 Preliminaries

In this section we recall the notions of qualitative calculi and their weak composition tables. Interested readers may consult e.g. [17,11] for more information.

Definition 1. Suppose U is a universe of spatial or temporal entities, and \mathcal{B} is a set of jointly exhaustive and pairwise disjoint (JEPD) binary relations on U . We call the Boolean algebra generated by \mathcal{B} a qualitative calculus, and call relations in \mathcal{B} the basic relations of this qualitative calculus.

We consider a simple example.

Example 1 (Point Algebra). Suppose $U = \mathbb{R}$. For two points a, b in U , we have either $a < b$, or $a = b$, or $a > b$. Let $\mathcal{B} = \{<, =, >\}$. Then \mathcal{B} is a JEPD set of relations on U . We call the Boolean Algebra generated by \mathcal{B} the Point Algebra.

We next recall the central notion of weak composition.

Definition 2. Suppose \mathcal{M} is a qualitative calculus on U , and \mathcal{B} is the set of its basic relations. The weak composition of two basic relations α and β in \mathcal{M} , denoted as $\alpha \circ_w \beta$, is defined as the smallest relation in \mathcal{M} which contains $\alpha \circ \beta$, the usual composition of α and β .

Usually, a qualitative calculus has a finite set of relations. The weak composition operation of \mathcal{M} can be summarised in an $n \times n$ table, where n is the cardinality of \mathcal{B} , and the cell specified by α and β contains all basic relations γ in \mathcal{B} such that $\gamma \cap \alpha \circ \beta \neq \emptyset$. The CT of the Point Algebra is given in Table 1.

Table 1. The CT of the Point Algebra, where $*$ is the universal relation

| | | | |
|---------|-----|-----|-----|
| \circ | $<$ | $=$ | $>$ |
| $<$ | $<$ | $<$ | $*$ |
| $=$ | $<$ | $=$ | $>$ |
| $>$ | $*$ | $>$ | $>$ |

Definition 3. Suppose \mathcal{M} is a qualitative calculus on U with basic relation set \mathcal{B} . For basic relations α, β, γ , we call $\langle \alpha, \gamma, \beta \rangle$ a composition triad, or c-triad, if $\gamma \subseteq \alpha \circ_w \beta$.

We can determine if a 3-tuple is a c-triad as follows (cf. Fig. 1).

Proposition 1. A 3-tuple $\langle \alpha, \gamma, \beta \rangle$ of basic relations in \mathcal{M} is a c-triad iff $\gamma \cap \alpha \circ \beta \neq \emptyset$, which is equivalent to saying that the basic constraint network

$$\{x\alpha y, y\beta z, x\gamma z\} \quad (2)$$

is consistent, i.e. it has a solution in U .

To compute the weak composition of α and β , one straightforward method is to find all basic relations γ such that $\langle \alpha, \gamma, \beta \rangle$ is a c-triad.

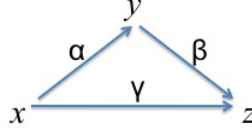


Fig. 1. A c-triad $\langle \alpha, \gamma, \beta \rangle$

3 A General Method for Computing CT

In this section, we propose a general approach to compute the composition table of a qualitative calculus \mathcal{M} with domain U and basic relation set \mathcal{B} . The approach is based on the observation that each triple of objects in U derives a valid c-triad.

Proposition 2. *Suppose a, b, c are three objects in U . Then $\langle \rho(a, b), \rho(a, c), \rho(b, c) \rangle$ is a c-triad, where $\rho(x, y)$ is the basic relation in \mathcal{M} that relates x to y .*

It is clear that six (different or not) c-triads can be generated if we consider all permutations of a, b, c .

To compute the CT of \mathcal{M} , the idea is to choose randomly a triple of elements in U and then compute and record the c-triads related to these objects in a dynamic table. Continuing in this way, we will get more and more c-triads until the dynamic table becomes stable after sufficient large loops. The basic algorithm is given in Algorithm 1, where D is a subdomain of U , Ψ decides when the procedure terminates, TRIAD records the number of c-triads obtained when the procedure terminates, and LASTFOUND records the time when the last triad is first recorded. For a calculus with unknown CT, the condition may be assigned with the form $\text{LOOP} \leq 1,000,000$ (i.e., the algorithm loops one million times), or $\text{LOOP} \leq \text{LASTFOUND} + 100,000$ (i.e., until no new c-triad is found in the last one hundred thousand loops), or their conjunction. If the CT is known and we want to double-check it, then the boundary condition could be set to $\text{TRIAD} < N$ to save time, where N is the number of c-triads of the calculus.

We make further explanations here.

Suppose \mathcal{M} is a qualitative calculus on U . Recall U is often an infinite set. We need first to decide a finite subdomain D of U , as computers only deal with numbers with finite precision. Once D is chosen, we run the loop, say, one million times. Therefore, one million instances of triples of elements in D are generated. We then record all computed c-triads in a dynamic table. It is reasonable to claim that the table is stable if no new entry has been recorded after a long time (e.g. as long as the time has past to get all recorded c-triads). Because D is finite, Algorithm 1 will generate a stable table after a sufficient large number of iterations.

We observe that a finite subdomain D may restrict the possible c-triads if it is selected inappropriately. We introduce a notion to characterise the appropriateness of a subdomain.

Definition 4. *Suppose \mathcal{M} is a qualitative calculus defined on the universe U . A nonempty subset D of U is called a 3-complete subdomain of \mathcal{M} if each consistent basic network as specified in Eq. 2 has a solution in D .*

Algorithm 1: Computing the Composition Table of \mathcal{M}

Input: A subdomain D of \mathcal{M} , and a boundary condition Ψ related to \mathcal{M}
Output: The Composition Table CT of \mathcal{M}
 Initialise CT ;
 LOOP \leftarrow 0;
 TRIAD \leftarrow 0;
 LASTFOUND \leftarrow 0;
while Ψ **do**
 LOOP \leftarrow LOOP + 1;
 Generate triple of objects $(a, b, c) \in D^3$ randomly;
 $\alpha \leftarrow$ the basic relation between a and b ;
 $\beta \leftarrow$ the basic relation between b and c ;
 $\gamma \leftarrow$ the basic relation between a and c ;
 $\alpha' \leftarrow$ the basic relation between b and a ;
 $\beta' \leftarrow$ the basic relation between c and b ;
 $\gamma' \leftarrow$ the basic relation between c and a ;
 for $\langle r, s, t \rangle \in \{\langle \alpha, \gamma, \beta \rangle, \langle \alpha', \beta, \gamma \rangle, \langle \gamma, \alpha, \beta' \rangle, \langle \beta, \alpha', \gamma' \rangle, \langle \beta', \gamma', \alpha' \rangle, \langle \gamma', \beta', \alpha \rangle\}$
 do
 if $\langle r, s, t \rangle$ is not in CT **then**
 Record triad $\langle r, s, t \rangle$ to CT ;
 TRIAD \leftarrow TRIAD + 1;
 LASTFOUND \leftarrow LOOP;
 end
 end
end
return CT .

If D is a 3-complete subdomain, then, for each c-triad $\langle \alpha, \gamma, \beta \rangle$, there are a, b, c in D such that $(a, b) \in \alpha$, $(b, c) \in \beta$, and $(a, c) \in \gamma$. Therefore, to determine the CT of \mathcal{M} , we need only consider instances of triples in D .

Note that no matter whether the subdomain D is 3-complete, the algorithm always generates ‘valid’ triads, in the sense that any 3-tuple $\langle \alpha, \gamma, \beta \rangle$ in the CT generated is indeed a c-triad of the calculus. However, the algorithm only converges to the correct CT when the subdomain D is 3-complete.

It is of course important questions to find 3-complete subdomains or to decide if a particular subdomain is 3-complete. However, it seems that there is no general answer for arbitrary qualitative calculi, since the questions are closely related to the semantics of the calculi. For a particular calculus, e.g. IA, this can be verified by formal analysis. Note that a superset of a 3-complete subdomain is also 3-complete. To make sure a chosen subdomain D is 3-complete, we often apply the algorithm on several of its supersets at the same time. If the same number is generated for all subdomains, we tend to believe that D is 3-complete and the generated table is the CT of \mathcal{M} . Note a formal proof is necessary to guarantee the 3-completeness of D .

Even if a CT of \mathcal{M} has been somehow obtained, our method can be used to verify its correctness. Double-checking is necessary since computing the CT is error-prone (see the last paragraph of page 1). If there is a c-triad that does not appear in the previously

given table, something must be wrong with the table, because the c-triads computed by Algorithm 1 are always valid. It is also possible that the algorithm terminates with a fragment of given composition table. We then can make theoretical analysis to see if the missing c-triads are caused by the incompleteness of the subdomain. If so, we modify the subdomain and run the algorithm again, otherwise, the missing c-triads are likely to be invalid c-triads.

Another thing we should keep in mind is how to generate a triple of elements (a, b, c) from D . Note that if D is small (e.g. in the cases of PA and IA), we can generate all possible triples. If D contains more than 1000 elements, then it will be necessary to generate the triples randomly as there are over a billion different triples. The distribution over D may affect the efficiency of the algorithm. Assuming that we have very limited knowledge of the calculus \mathcal{M} , it is natural to take a, b and c independently with respect to the uniform distribution. We note that the better we understand the calculus, the more appropriate the distribution we may choose.

To increase the efficiency of the algorithm, we sometimes use the algebraic properties of the calculus. For example, if the identity relation id is a basic relation, then by $\alpha \circ_w id = \alpha = id \circ_w \alpha$ and $id \subseteq \alpha \circ_w \alpha^\sim$, we need not compute the c-triads involving id , where α^\sim is the converse of α . This is to say, the algorithm only needs to generate pairwise different elements. As another example, suppose that the calculus is closed under converse, i.e. the converse of a basic relation is still a basic relation. Then in Algorithm 1 we need only compute α, β, γ . The other relations and c-triads can be obtained by replacing α', β', γ' in the algorithm by, respectively, $\alpha^\sim, \beta^\sim, \gamma^\sim$. Similar results have been reported in [18].

In the following we examine three important examples. All experiments were conducted on a 3.16 GHZ Intel Core 2 Duo CPU with 3.25 GB RAM running Windows XP. Note the results rely on the random number generator. As our aim is to show the feasibility of the algorithm rather than investigating the efficiency issues, we only provide one group of the results and do not make any statistical analysis.

3.1 The Interval Algebra and the INDU Calculus

We start with the best known qualitative calculus.

Example 2 (Interval Algebra). Let U be the set of closed intervals on the real line. Thirteen binary relations between two intervals $x = [x^-, x^+]$ and $y = [y^-, y^+]$ are defined in Table 2. The Interval Algebra [2] is the Boolean algebra generated by these thirteen JEPD relations.

The CT for IA has been computed in 1983 in Allen's famous work. When applying Algorithm 1 to IA, we do not consider all intervals. Instead, we restrict the domain to the set of all intervals contained in $[0, M)$ that have integer nodes

$$D_M = \{[p, q] | p, q \in \mathbb{Z}, 0 \leq p < q < M\},$$

and use uniform distribution to choose random intervals. It is easy to see that the size of the domain is $M(M-1)/2$. Note that to converge fast and generate all entries, we need to choose an appropriate M . If M is too small, then it is possible that some c-triads can

Table 2. Basic IA relations and their converses, where $x = [x^-, x^+]$, $y = [y^-, y^+]$ are two intervals.

| Relation | Symbol | Converse | Meaning |
|----------|-----------|-----------|-------------------------|
| before | b | bi | $x^- < x^+ < y^- < y^+$ |
| meets | m | mi | $x^- < x^+ = y^- < y^+$ |
| overlaps | o | oi | $x^- < y^- < x^+ < y^+$ |
| starts | s | si | $x^- = y^- < x^+ < y^+$ |
| during | d | di | $y^- < x^- < x^+ < y^+$ |
| finishes | f | fi | $y^- < x^- < x^+ = y^+$ |
| equals | eq | eq | $x^- = y^- < x^+ = y^+$ |

Table 3. Implementation for IA, where TRIAD is the number of c-triads recorded by running the algorithm on D_M for $M = 4$ to $M = 20$, LASTFOUND is the loop when the last triad is first recorded

| M | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|-----------|-------|-------|------|------|------|-------|------|-------|-----|
| TRIAD | 139 | 319 | 409 | 409 | 409 | 409 | 409 | 409 | 409 |
| LASTFOUND | 92 | 629 | 1501 | 878 | 2111 | 3517 | 728 | 697 | 932 |
| M | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
| TRIAD | 409 | 409 | 409 | 409 | 409 | 409 | 409 | 409 | |
| LASTFOUND | 11212 | 20249 | 7335 | 4343 | 3632 | 17862 | 5533 | 43875 | |

not be instantiated. On the other hand, if M is too big, relations that require one or more exact matches (such as **m** in IA and **m⁼** in the INDU calculus to be introduced in the next example) is very hard to generate, i.e. the probability of generating such an instance is very small. For a new qualitative calculus, there is no general rules for choosing M . Usually, pilot experiments are necessary to better understand the characteristics of the calculus.

Table 3 summarises the results for $M = 4$ to $M = 20$. In the experiment, we generate one million instances of triples of elements for each domain D_M . In all cases the dynamic table becomes stable in less than 50,000 loops. When the table becomes stable, the numbers of triads computed are not always the correct one (that is 409). This is mainly because the domain is too small. For M bigger than or equal to six, we always get the correct number of triads.² The loops needed (i.e. LASTFOUND) vary from less than a thousand to more than 43 thousand (see Table 3). In general, the smaller the domain is the more efficient the algorithm is.

Example 3 (INDU calculus). The INDU calculus [5] is a refinement of IA. For each pair of intervals a, b , INDU allows us to compare the durations of a, b . This means, some IA relations may be split into three sub-relations. For example, **b** is split into three relations **b[<]**, **b⁼**, **b[>]**. Similar situations apply to **m**, **o**, **oi**, **mi**, and **bi**. The other seven relations have no proper sub-relations. Therefore, INDU has 25 basic relations.

INDU is quite unlike IA. For example, it is not closed under composition, and a path-consistent basic network is not necessarily consistent [19].

² The 3-completeness of D_6 follows from the fact that each consistent IA network involving three variables has a solution in D_6 .

Table 4. Implementation for INDU, where TRIAD is the number of c-triads recorded by running the algorithm on D_M for $M = 6$ to $M = 20$, LASTFOUND is the loop when the last triad is first recorded

| M | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|-----------|-------|-------|-------|-------|-------|-------|-------|-------|
| TRIAD | 1045 | 1531 | 1819 | 1987 | 2041 | 2053 | 2053 | 2053 |
| LASTFOUND | 3766 | 5753 | 10417 | 35201 | 35891 | 25031 | 12512 | 27728 |
| M | 14 | 15 | 16 | 17 | 18 | 19 | 20 | |
| TRIAD | 2053 | 2053 | 2053 | 2053 | 2053 | 2053 | 2053 | |
| LASTFOUND | 17223 | 24578 | 14758 | 22491 | 29034 | 49693 | 19772 | |

Applying our algorithm to INDU, we use the same subdomain D_M as for IA. From Table 4 we can see that D_6 is no longer 3-complete: more than 1000 c-triads do not appear in the stable table. The table becomes complete in D_{11} , which has 2053 c-triads. The 3-completeness of D_{11} is confirmed by the following proposition.

Proposition 3. *The INDU calculus has at most 2053 c-triads.*

Proof (Sketch). For any three INDU relations $\alpha^{\star_1}, \beta^{\star_2}, \gamma^{\star_3}$ ($\star_1, \star_2, \star_3 \in \{<, =, >\}$), it is easy to see that $\langle \alpha^{\star_1}, \gamma^{\star_2}, \beta^{\star_3} \rangle$ is a valid c-triad of INDU only if $\langle \alpha, \gamma, \beta \rangle$ is a valid c-triad of IA and $\langle \star_1, \star_2, \star_3 \rangle$ is a valid c-triad of PA. We note that for IA relations in $\{d, s, f, eq, si, fi, di\}$, only $d^<, s^<, f^<, eq^=, si^>, fi^>, di^>$ are valid INDU relations. It is routine to check that there are only 2053 triples of INDU relations that satisfy the above two constraints. We recall that IA has 409 c-triads (see Table 3), and PA has 13 c-triads (see Table 1). \square

Since 2053 valid c-triads are recorded by running the algorithm on D_{11} for INDU, we know INDU has precisely 2053 c-triads, and D_{11} is 3-complete for INDU. Moreover, we have that $\langle \alpha^{\star_1}, \gamma^{\star_2}, \beta^{\star_3} \rangle$ is a valid c-triad of INDU *if and only if* $\langle \alpha, \gamma, \beta \rangle$ is a valid c-triad of IA and $\langle \star_1, \star_2, \star_3 \rangle$ is a valid c-triad of PA.

It seems that this is the first time that the CT of INDU has been computed.

3.2 The Oriented Point Relation Algebra

In the \mathcal{OPRA}_m calculus, where m is a parameter characterizing its granularity, each object is represented as an oriented point (*o-point* for short) in the plane. Each o-point has an orientation. Based on which, $2m - 1$ other directions are introduced according to the chosen granularity. Any other o-point is located on either a ray or in a section between two consecutive rays. Each of these rays and sections is assigned an integer from 0 to $4m - 1$. The relative directional information of two o-points A, B is uniquely encoded in a pair of integer numbers (s, t) , where s is the ray or section of A in which B is located, and t is the ray or section of B in which A is located. Such a relation is also written as $A_m \angle_s^t B$. In the case that the locations of A and B coincide, the relation between A and B is written as $_m \angle_s B$, where s is the ray or section of A in which the orientation of B is located. Therefore, there are $4m(4m + 1)$ basic relations in \mathcal{OPRA}_m .

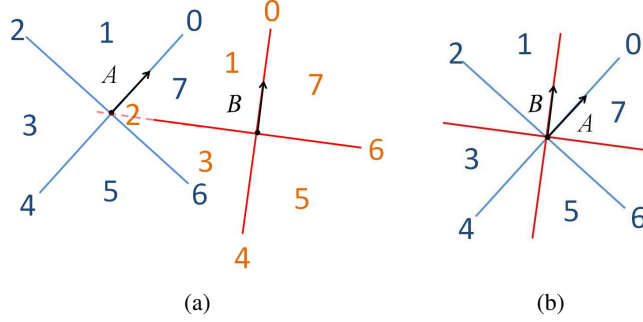


Fig. 2. Two o-points A, B with the \mathcal{OPRA}_2 relation (a) ${}_2\angle_7^2$ and (b) ${}_2\angle_1$.

There are two natural ways to represent o-points. One uses the Cartesian coordinate system, the other use polar coordinate system. We next show the choice of coordinate system will significantly affect the experimental results, which are compared with that of [14].

In the Cartesian coordinate system, an o-point P is represented by its coordination (x, y) and its orientation ϕ .

Definition 5. Let M_1 and M_2 be two positive integers. We define a Cartesian based subdomain of \mathcal{OPRA}_m as

$$D_c(M_1, M_2) = \{((x, y), \phi) : x, y \in [-M_1, M_1] \cap \mathbb{Z}, \phi \in \Phi_{M_2}\},$$

where $\Phi_{M_2} \equiv \{0, 2\pi/M_2, \dots, (M_2 - 1)/M_2 \times 2\pi\}$.

Table 5. Implementation for \mathcal{OPRA}_1 on a Cartesian coordinated domain $D_c(M_1, M_2)$, where TRIAD is the number of c-triads computed by running the algorithm on $D_c(M_1, M_2)$ for $M_1 = 6$; LASTFOUND is the loop when the last triad is first recorded for $M_2 = 8$ (in the 2nd last row) and $M_2 = 16$ (in the last row)

| M_2 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | 12 | 16 |
|--------------------------|-------|------|--------|------|--------|-------------|-----------|------|-------------|
| TRIAD | 148 | 1024 | 1056 | 1024 | 1024 | 1440 | 1024 | 1408 | 1440 |
| M_1 | 2 | | 4 | | 6 | | 8 | | 10 |
| LASTFOUND ($M_2 = 8$) | 8082 | | 35932 | | 411893 | | 881787 | | > 1000000 |
| LASTFOUND ($M_2 = 16$) | 18618 | | 295936 | | 174490 | | > 1000000 | | > 1000000 |

Our experimental results show that, for \mathcal{OPRA}_1 , the algorithm converges and generates the correct CT for subdomains with $M_1 \geq 2$ and $M_2 \in \{8, 16\}$. That is, the smallest 3-complete subdomain is $D_c(2, 8)$.

For \mathcal{OPRA}_2 , however, the algorithm does not compute the desired CT in ten million loops. Actually, it is impossible to compute the desired CT if we use Cartesian coordination. Consider the following example. Suppose A, B, C are three o-points, such

that $\triangle ABC$ is an acute triangle, and the orientation of A is the same as the direction from A to B , the orientations of B and C are similar. In this configuration, we have $A_2\angle_0^1B$, $B_2\angle_0^1C$, and $A_2\angle_1^0C$. This configuration, however, cannot be realised in a Cartesian based subdomain.³

Table 6. Implementation for \mathcal{OPRA}_2 on a Cartesian coordinated domain $D_c(M_1, M_2)$, where TRIAD is the number of c-triads computed by running Algorithm 1 ten million times on $D_c(M_1, M_2)$ for $M_1 = 6$

| M_2 | 2 | 4 | 6 | 8 | 10 | 12 | 16 |
|-------|------|------|-------|-------|-------|-------|-------|
| TRIAD | 2704 | 2704 | 21792 | 23616 | 21792 | 21792 | 35232 |

Based on the above observation, we turn to the polar coordinated representation. In the polar coordinate system, an o-point P is represented by its polar coordination (ρ, θ) and its orientation ϕ .

Definition 6. Let M_1 and M_2 be two positive integers. We define a polar coordinated subdomain of \mathcal{OPRA}_m as

$$D_p(M_1, M_2) = \{((\rho, \theta), \phi) : \rho \in [0, M_1] \cap \mathbb{Z}, \theta, \phi \in \Phi_{M_2}\},$$

where $\Phi_{M_2} \equiv \{0, 2\pi/M_2, \dots, (M_2 - 1)/M_2 \times 2\pi\}$.

As in Cartesian based subdomains, the parameter M_2 determines if a domain is complete, while M_1 determines the efficiency of the algorithm. For \mathcal{OPRA}_1 , we have $D(M_1, M_2)$ is a 3-complete subdomain if $M_1 \geq 2$ and $M_2 = 6, 8, 10, 12, 16$ (see Table 7); for \mathcal{OPRA}_2 , we have $D(M_1, M_2)$ is 3-complete if $M_1 \geq 4$ and $M_2 = 6, 10, 12, 16$ (see Table 8).

Table 7. Implementation for \mathcal{OPRA}_1 on a polar coordinated domain $D_p(M_1, M_2)$, where TRIAD is the number of c-triads computed by running the algorithm on $D_p(M_1, M_2)$ for $M_1 = 6$; LASTFOUND is the loop when the last triad is first recorded for $M_2 = 8$ (in the 2nd last row) and $M_2 = 16$ (in the last row)

| M_2 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | 12 | 16 |
|--------------------------|----|------|------|------|-------------|-------------|-------------|-------------|-------------|
| TRIAD | 52 | 1024 | 1032 | 1408 | 1440 | 1440 | 1440 | 1440 | 1440 |
| M_1 | | | | | 4 | 6 | 8 | 10 | 16 |
| LASTFOUND ($M_2 = 8$) | | | | | 3072 | 4868 | 22327 | 10363 | 38843 |
| LASTFOUND ($M_2 = 16$) | | | | | 26219 | 45831 | 121542 | 71205 | 146536 |

3.3 The Region Connection Calculus

Our algorithm works very well for simple objects like points and intervals. We next consider a region-based topological calculus RCC-8. It is worth noting that an auto-

³ The proof of this statement is much involved and omitted in this paper.

Table 8. Implementation for \mathcal{OPRA}_2 on a polar coordinated domain $D_p(M_1, M_2)$, where TRIAD is the number of c-triads computed by running the algorithm on $D_p(M_1, M_2)$ for $M_1 = 6$

| M_2 | 2 | 3 | 4 | 6 | 8 | 10 | 12 | 16 |
|-------|-----|-------|------|--------------|-------|--------------|--------------|--------------|
| TRIAD | 400 | 24672 | 2128 | 36256 | 23616 | 36256 | 36256 | 36256 |

mated derivation of the composition table was reported in [20] for a similar calculus (the 9-intersection model).

Example 4 (RCC-8 algebra). Let U be the set of bounded plane regions (i.e. nonempty regular closed sets in the plane). Five binary relations are defined in Table 9. The RCC-8 algebra [4] is the Boolean algebra generated by these five relations, the identity relation **EQ**, and the converses of **TPP** and **NTPP**.

Table 9. A topological interpretation of basic RCC-8 relations in the plane, where a, b are two bounded plane regions, and a°, b° are the interiors of a, b , respectively.

| Relation | Meaning |
|-------------|---|
| DC | $a \cap b = \emptyset$ |
| EC | $a \cap b \neq \emptyset, a^\circ \cap b^\circ = \emptyset$ |
| PO | $a \not\subseteq b, b \not\subseteq a, a^\circ \cap b^\circ \neq \emptyset$ |
| TPP | $a \subset b, a \not\subset b^\circ$ |
| NTPP | $a \subset b^\circ$ |

Table 10. Implementation for RCC-8, where TRIAD is the number of c-triads computed by running the algorithm on D_M using rectangles, LASTFOUND is the loop when the last triad is first recorded

| M | 4 | 5 | 6 | 8 | 10 | 15 | 20 |
|-----------|-------|------|---------|-------|--------|--------|---------|
| TRIAD | 114 | 177 | 192 | 192 | 192 | 192 | 192 |
| LASTFOUND | 14776 | 6513 | 2332646 | 56067 | 198255 | 261729 | 1521173 |

Plane regions are much more complicated to represent than intervals or o-points. In most cases they are approximated by polygons or digital regions (i.e., a subset of \mathbb{Z}^2). Furthermore, it is natural to take a shot on simple objects at the beginning, since they are easy to deal with and important in applications. For RCC-8, we make experiments over two subdomains: rectangles and disks. The experiments show that these subdomains are good enough for our purpose, but when necessary, we could also consider general polygons or bounded digital regions.

We first consider subdomains whose elements are rectangles sides of which are parallel to the two axes. We introduce one parameter M , and require the four nodes be points in $[0, M) \times [0, M) \cap \mathbb{Z}^2$. The complete RCC-8 CT has 193 table entries.

Since $\mathbf{EQ} \circ \mathbf{EQ} = \mathbf{EQ}$, we know $\langle \mathbf{EQ}, \mathbf{EQ}, \mathbf{EQ} \rangle$ is a c-triad. The other 192 c-triads can be confirmed using our algorithm. In Table 10, we show the results of running the algorithm 10 million times and require M vary from 4 to 20. We can see from the table that D_M is a 3-complete subdomain only if $M \geq 6$.

Table 11. Implementation for RCC-8, where TRIAD is the number of c-triads computed by running the algorithm on D_M using disks, LASTFOUND is the loop when the last triad is first recorded

| M | 4 | 5 | 6 | 8 | 10 | 15 | 20 |
|-----------|------|------|------|-------|--------|--------|---------|
| TRIAD | 188 | 192 | 192 | 192 | 192 | 192 | 192 |
| LASTFOUND | 1759 | 8913 | 9489 | 25955 | 113757 | 942914 | 2961628 |

We next consider subdomains consisting of disks (see Table 11). We introduce one parameter M , and require $x, y \in [0, M] \cap \mathbb{Z}$, $r \in [1, M] \cap \mathbb{Z}$, where (x, y) and r are, respectively, the centre and the radius of the closed disk $B((x, y), r)$. In this case, $M = 5$ is good enough to generate all c-triads. We notice that the number of loops needed (i.e. LASTFOUND) increases quickly as M increases. For example, when $M = 20$, the dynamic table becomes stable after nearly 3 million loops. This is mainly due to that an instance of the c-triad $\langle \mathbf{NTPP}, \mathbf{NTPP}, \mathbf{NTPP} \rangle$ is very hard to generate. The ‘hard’ c-triad is, however, easy to prove.

4 Further Discussions

In the last section, we have applied our algorithm to generate the CTs of IA, INDU, \mathcal{OPRA}_1 , \mathcal{OPRA}_2 , and RCC-8. In this section, we discuss the advantages and disadvantages of our method. The algorithm works very well for simple objects like points, intervals, rectangles, and disks, especially in a small subdomain. For a qualitative calculus with less than 100 basic relations, it can compute the CT in a few minutes.

We also considered larger calculi. The Oriented Point Relation Algebras \mathcal{OPRA}_3 and \mathcal{OPRA}_4 have, respectively, 156 and 272 basic relations. Applying our algorithm to an appropriate polar coordinated subdomain $D_p(M_1, M_2)$, 261,576 and 1,082,752 c-triads, respectively, have been found in a few hours, which coincide with those computed in [14]. This implies that the corresponding subdomains are 3-complete.

For calculi defined over regions, the main obstacle of using our approach is the cost of generating random regions. For RCC-8, we circumvent this obstacle by taking rectangles and disks. But this circumvention does not work for the Cardinal Direction Calculus (CDC) [7], as the shape of the region matters in this calculus. The CDC contains 218 basic relations. We run our algorithm for the CDC on the subdomain containing digital regions in $[0, 5] \times [0, 5]^4$, using normal distribution. The result is not ideal. After one day, we have computed about 60% of the total 2.2 million c-triads of the CDC. Improvements will be made later, adopting more appropriate or heuristic distribution.

⁴ The 3-completeness of this subdomain is confirmed by results reported in [16].

In many applications of qualitative calculi, the objects used are often restricted. Take \mathcal{OPRA}_2 as example. In many real world applications, e.g. the Interstate Highway System of the USA, oriented objects are all taken from a underlying graph. In these cases, each o-point has only a few possible directions. To support reasoning with this domain, we had better have a *customised* \mathcal{OPRA}_2 calculus, together with a customised CT. Our algorithm works perfect to this end. For example, consider the restriction of \mathcal{OPRA}_2 calculus on \mathbb{Z}^2 . Each o-point in this calculus has integer coordinations and has one of the four orientations from $\{0, \pi/2, \pi, 3\pi/2\}$. Using our algorithm, the CT for this customised calculus has been generated in a few minutes. Experiment result shows that this customised calculus has 2704 c-triads.

Even for the well-known Interval Algebra, our algorithm suggests a new viewpoint for efficient reasoning. We note that, based on our method, we can easily compute the *probability* of each basic relation in the weak composition of any two basic IA relations. This may be used in approximate temporal reasoning, especially when the application domain has a different structure than the universe of IA. Work towards this direction will be reported in another paper.

5 Conclusion

In this paper, we introduced a general and simple semi-automatic method for computing the composition tables of qualitative calculi. The described method is a very natural approach, and similar idea was used to derive composition tables for an elaboration of RCC with convexity [21], and for a ternary directional calculus [22]. The table computed in [21] was acknowledged there as incomplete. The table computed in [22] is complete, but its completeness was guaranteed by manually checking all geometric configurations that satisfy the table. Except these two works, very little attention has been given to this natural approach in the literature on composition tables. We think a systematic examination is necessary to discover both the strong and weak points of this approach.

We implemented the basic algorithm for several well-known qualitative calculi, including the Interval Algebra, INDU, \mathcal{OPRA}_m for $m = 1 \sim 4$, and RCC-8. Our experiments suggest that the proposed method works very well for point-based calculi, but not so well for region-based calculi. In particular, we established, as far as we know, for the first time the correct CT for INDU, and confirmed the validity of the algorithm reported for the \mathcal{OPRA} calculi [14]. Our method can be easily integrated into existing qualitative solvers e.g. SparQ [8] or GQR [9]. This provides a partial answer to the challenge proposed in [12].

Recently, Wolter proposes (in an upcoming article [23]) to derive composition tables by solving systems of polynomial (in)equations over the reals. This approach works well for several point-based calculi, but not always generates the complete composition table.

Our method relies on the assumption that the qualitative calculus has a small ‘discretised’ 3-complete subdomain. All calculi considered in this paper satisfy this property. It is still open whether all interesting calculi appeared in the literature satisfy this prop-

erty. Future work will also discuss the applications of our method for reasoning with a customised composition table.

References

1. Allen, J.F.: An interval-based representation of temporal knowledge. In Hayes, P.J., ed.: IJCAI, William Kaufmann (1981) 221–226
2. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* **26**(11) (1983) 832–843
3. Vilain, M.B., Kautz, H.A.: Constraint propagation algorithms for temporal reasoning. In: AAAI. (1986) 377–382
4. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In: KR. (1992) 165–176
5. Pujari, A.K., Kumari, G.V., Sattar, A.: INDU: An interval and duration network. In Foo, N.Y., ed.: Australian Joint Conference on Artificial Intelligence. Volume 1747 of Lecture Notes in Computer Science., Springer (1999) 291–303
6. Moratz, R.: Representing relative direction as a binary relation of oriented points. In Brewka, G., Coradeschi, S., Perini, A., Traverso, P., eds.: ECAI. Volume 141 of Frontiers in Artificial Intelligence and Applications., IOS Press (2006) 407–411
7. Goyal, R.K.: Similarity assessment for cardinal directions between extended spatial objects. PhD thesis (2000) The University of Maine.
8. Wallgrün, J.O., Frommberger, L., Wolter, D., Dylla, F., Freksa, C.: Qualitative spatial representation and reasoning in the sparq-toolbox. In Barkowsky, T., Knauff, M., Ligozat, G., Montello, D.R., eds.: Spatial Cognition. Volume 4387 of Lecture Notes in Computer Science., Springer (2006) 39–58
9. Westphal, M., Wöfl, S., Gantner, Z.: GQR: A fast solver for binary qualitative constraint networks. In: AAAI Spring Symposium on Benchmarking of Qualitative Spatial and Temporal Reasoning Systems. (2009)
10. Li, S., Ying, M.: Region connection calculus: Its models and composition table. *Artif. Intell.* **145**(1-2) (2003) 121–146
11. Renz, J., Ligozat, G.: Weak composition for qualitative spatial and temporal reasoning. In van Beek, P., ed.: CP. Volume 3709 of Lecture Notes in Computer Science., Springer (2005) 534–548
12. Cohn, A.G.: The challenge of qualitative spatial reasoning. *ACM Comput. Surv.* **27**(3) (1995) 323–325
13. Frommberger, L., Lee, J.H., Wallgrün, J.O., Dylla, F.: Composition in \mathcal{OPRA}_m . Technical report, SFB/TR 8, University of Bremen (2007)
14. Mossakowski, T., Moratz, R.: Qualitative reasoning about relative direction on adjustable levels of granularity. *CoRR* **abs/1011.0098** (2010)
15. Skiadopoulos, S., Koubarakis, M.: Composing cardinal direction relations. *Artif. Intell.* **152**(2) (2004) 143–171
16. Liu, W., Zhang, X., Li, S., Ying, M.: Reasoning about cardinal directions between extended objects. *Artif. Intell.* **174**(12-13) (2010) 951–983
17. Ligozat, G., Renz, J.: What is a qualitative calculus? a general framework. In Zhang, C., Guesgen, H.W., Yeap, W.K., eds.: PRICAI. Volume 3157 of Lecture Notes in Computer Science., Springer (2004) 53–64
18. Bennett, B.: Some observations and puzzles about composing spatial and temporal relations. In: in R. Rodríguez (ed.), Proceedings ECAI-94 Workshop on Spatial and Temporal Reasoning. (1994) 65–72

19. Balbiani, P., Condotta, J.F., Ligozat, G.: On the consistency problem for the INDU calculus. *J. Applied Logic* **4**(2) (2006) 119–140
20. Egenhofer, M.J.: Deriving the composition of binary topological relations. *J. Vis. Lang. Comput.* **5**(2) (1994) 133–149
21. Cohn, A.G., Randell, D.A., Cui, Z., Bennett, B.: Qualitative spatial reasoning and representation. In Carreté, N.P., Singh, M.G., eds.: *Qualitative Reasoning and Decision Technologies*, Barcelona, CIMNE (1993) 513–522
22. Clementini, E., Skiadopoulos, S., Billen, R., Tarquini, F.: A reasoning system of ternary projective relations. *IEEE Trans. on Knowl. and Data Eng.* **22** (February 2010) 161–178
23. Wolter, D.: Analyzing qualitative spatio-temporal calculi using algebraic geometry. *Spatial Cognition & Computation* (to appear).